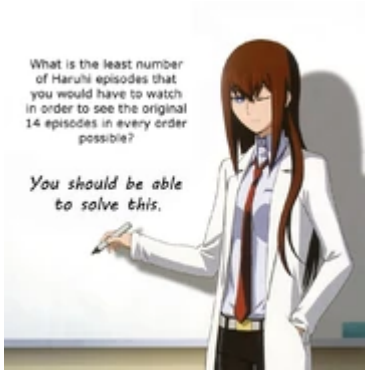
The Haruhi Problem

"What is the least number of Haruhi episodes that you would have to watch in order to see the original 14 episodes in every order possible?"

Or, more formally, "what is the shortest string containing all permutations of a set of n elements?"



What is the smallest string containing all permutations of a set of n elements for n=14?

Contents [show]

The Problem Edit

- You have an n episode TV series. You want to watch the episodes in every order possible. What is the least number of episodes that you would have to watch?
- Overlapping is allowed. For example, in the case of n=2, watching episode 1, then 2, then 1 again, would fit the criteria.
- The orders must be continuous. For example, (1,2,1,3) does NOT contain the sequence (1,2,3)

Current Collaboration Edit

Algorithm and Bounds Edit

Credit: Anonymous (/sci/)

4 permutations are contained within each n-cycle. So the goal of this algorithm is to systematically generate the sequence, and show that any other method would give a less efficient method. In the long run, I'd like to create some axioms/theorems for the proof, such as more overlap=more efficiency. I think this will call for some modular arithmetic to generalize it for all n but I'm not sure how to do so.

by follow, I mean use it as a "rule" to tell you the next number in the sequence

Start with 1 and follow (1 2 3 4) 6 times. We do this by convention. 1,[2,3,4,1,2,3]

Follow (1 4 2 3) 5 times. This group must be chosen, as (23) is in it. 1,2,3,4,1,2,3,[1,4,2,3,1]

Follow (1 2 4 3) 5 times. This is the last group with (3 1) in it. 1,2,3,4,1,2,3,1,4,2,3,1,[2,4,3,1,2]

Since no remaining groups have (1 2) in them, we have to choose one

A): Following (1 4 3 2) 6 times would end the sequence in wrong: 1,2,3,4,1,2,3,1,4,2,3,1,2,4,3,1,2[1,4,3,2,1,4] No group left has (1 4) in it, so this option looses efficiency

B): Following (1 3 2 4) 6 times would end the sequence in wrong: 1,2,3,4,1,2,3,1,4,2,3,1,2,4,3,1,2[1,3,2,4,3,2] No group left has (3 2) in it, so this option looses efficiency

C): Following (1 3 4 2) 6 times works: 1,2,3,4,1,2,3,1,4,2,3,1,2,4,3,1,2[1,3,4,2,1,3]

Follow (1 3 2 4) 5 times. This is the last group with (1 3) in it. 1,2,3,4,1,2,3,1,4,2,3,1,2,4,3,1,2,1,3,4,2,1,3,[2,4,1,3,2] Follow the last group (1 4 3 2) 5 times, to complete the sequence. 1,2,3,4,1,2,3,1,4,2,3,1,2,4,3,1,2,1,3,4,2,1,3,2,4,1,3,2,1,4,3,2,1

The Lower Bound Edit

I think I have a proof of the lower bound n! + (n-1)! + (n-2)! + n-3 (for

$n \geq 2$). I'll need to do this in multiple posts. Please look it over for any loopholes I might have missed.As in other posts, let n (lowercase) = the number of symbols; there are n! permutations to iterate through. The obvious lower bound is n! + n-1. We can obtain this as follows:

Let L = the running length of the string

N_0 = the number of permutations visited $X_0 = L - N_0$ When you write down the first permutation, X_0 is already n-1. For each new permutation you visit, the length of the string must increase by at least 1. So X_0 can never decrease. At the end, $N_0 = n!$, giving us $L \geq n! + n - 1$ .I'll use similar methods to go further, but first I'll need to explain my terminology...

Edges:

I'm picturing the ways to get from one permutation to the next as a directed graph where the nodes correspond to permutations and the edges to ways to get from one to the next. A k-edge is an edge in which you move k symbols from the beginning of the permutation to the end; for example,

1234567 -> 4567321

would be a 3-edge. Note that I don't include edges like

12345 -> 34512

in which you pass through through a permutation in the middle (in this case 23451). This example would be considered two edges:

12345 -> 23451 23451 -> 34512

From every node there is exactly one 1-edge, e.g.:

12345 -> 23451

These take you around in a cycle of length n. A 2-edge moves the first two symbols to the end. A priori, it could either reverse or maintain the order of those two symbols:

12345 -> 34521 12345 -> 34512

But the second, as already stated, is not counted as a 2-edge because it is a composition of two 1-edges. So there is exactly one 2-edge from every node.

1-loops:

I call the set of n permutations connected by a cyclic path of 1-edges a 1-loop. There are (n-1)! 1-loops.

The concept of 1-loops is enough to get the next easiest lower bound of n! + (n-1)! + n-2. That's because to pass from one 1-loop to another, it is necessary to take a 2-edge or higher. Let us define:

N_1 = the number of 1-cycles completed or that we are currently in $X_1 = L - N_1 - N_2$ The definition of N_1 is a bit more complicated than we need for this proof, but we'll need it later. You might ask, isn't N_1 just one more than the number of completed 1-cycles? No! When we have just completed a 1-cycle, it is equal to the number of completed 1-cycles. In order to increment N_1, we have to take a 2-edge, which increases L by 2 instead of 1. Therefore X_1 can never decrease. Since X_1 starts out with the value n-2, and we have to complete all (n-1)! 1-loops, we get the lower bound n! + (n-1)! + n-2.

2-loops:

Suppose we enter a 1-loop, iterate through all n nodes (as is done in the greedy palindrome algorithm), and then take a 2-edge out. The edge we exit by is determined by the entry point. The permutation that the 2-edge takes us to is determined by taking the entry point and rotating the first n-1 characters, e.g.:

12345 is taken by n-1 1-edges to 51234 which is taken by a 2-edge to 23415

If we repeat this process, it takes us around in a larger loop passing through n(n-1) permutations. I call this greater loop a 2-loop.

The greedy palindrome algorithm uses ever-larger loops; it connects (n-k+1) k-loops via (k+1)-edges to make (k+1)-loops. But I haven't been able to prove anything about these larger loops yet.

The tricky thing about 2-loops is that which 2-loop you're in depends on the point at which you entered the current 1-loop. Each of the n possible entry points to a 1-loop gives you a different 2-loop, so there are n*(n-2)! 2-loops, which overlap.

And now for the proof of the n! + (n-1)! + (n-2)! lower bound...

To review: n = alphabet length L = running string length

N_0 = number of permutations visited $X_0 = L - N_0$ N_1 = number of 1-cycles completed or that we are currently in $X_1 = L - N_1 - N_2$ In order to increase N_1, you must jump to a new 1-cycle -- having completed the one you are leaving. That means the next permutation P' in the 1-cycle (following your exit point P) is one you have already visited. Either you have at some point entered the 1-cycle at P', or this is the second or greater time you've visited P. If you have ever entered the 1-cycle at P', leaving at

P by a 2-edge will not take you to a new 2-cycle; you will be in the same 2-cycle you were in when you entered at P'.So these are the available ways to enter a 2-cycle you've never been in before:* take a 3-edge or higher

- take a 2-edge but don't increase N_1
- take a 2-edge from a permutation P that you were visiting for the second or greater time

In the first two cases, X_1 increases by 1 in the step under consideration. In the third case, X_1 must have increased by 1 in the previous step. Because of this third case, it is convenient to regard any series of edge traversals that takes you through permutations you've already visited as a single step. Then if we define N_2 = number of 2-cycles visited $X_2 = L - N_0 - N_1 - N_2$ the quantity X_2 does not decrease in any step. Since 2-cycles are n(n-1) long, you must visit at least (n-2)! 2-cycles. X_2 is initially n-3, giving us the lower bound $L \geq n! + (n-1)! + (n-2)! + n - 3$.

-Written by Anonymous

Resources and Information Edit

- http://pastebin.com/aNwANugC -Python algorithm by Anonymous
- http://www.notatt.com/permutations.pdf -proof for upper bound
- https://warosu.org/sci/thread/3751105 - source thread
- Latex'ed write up

Other Edit

- http://www.reddit.com/r/math/related/foj1l/the_shortest_string_containing_all_permutations/ http://oeis.org/A180632
- http://stackoverflow.com/questions/2253232/generate-sequence-with-all-permutations/2274978 http://forums.xkcd.com/viewtopic.php?f=17&t=68643

Hardmode Edit

Define H(n) as the number of sequences that are most efficient. for n=2, h(n)=2 {(1,2,1), (2,1,2)} what is the H(n)? Note that it simply isn't n!, for n=5, there are at least 2 efficient sequences that start with 12345

Retrieved from "https://mathsci.fandom.com/wiki/The_Haruhi_Problem?oldid=4132"

Categories: Add category